DOI: 10.3359/oz0802034 1.01: IZVIRNI ZNANSTVENI ČLANEK

članek

AVTORSKI PROCES V SPLETNEM OKOLJU

PRIMER SISTEMATIZACIJE DOPOLNILNE PROGRAMSKE OPREME

Maksimiljan Gerkeš

Institut informacijskih znanosti, Maribor

Kontaktni naslov:
maks.gerkes@izum.si

Izvleček

Čeprav je avtorski proces v spletnem okolju podprt z vrsto programskih pripomočkov, je lahko avtor postavljen v situacije, ki terjajo izdelavo dopolnilne programske opreme. Ker imajo te situacije lastnost ponovljivosti, se lahko odločimo za sistematizacijo dopolnilne programske opreme za reševanje izbranih kategorij problemov. S sistematizacijo želimo poiskati rešitev, ki bi omogočala reševanje sedanjih problemov, za katere smo že izdelali programe, in reševanje prihodnjih problemov. Prispevek obravnava oblikovanje več programskih sistemov, s katerimi rešujemo izbrane kategorije problemov v avtorskem procesu. Vsi podani sistemi so tudi dejansko izdelani in se uporabljajo v avtorskem procesu v spletnem okolju.

Ključne besede

avtorski proces, svetovni splet, ad hoc programska oprema, programska oprema, sistematizacija

Abstract

Although the authoring in web environment is supported by a range of program tools an author could find himself in situations that require the development of additional software. Because of repeatability of these situations the decision was made to systematize the additional software for solving authoring problems from the selected categories. With systematization it is intended to find a solution that would enable solving problems for which programs has already been made and future problems. This article describes the design of a few software systems that make possible to solve authoring problems from the selected categories. All the systems described are actually built and in use in web authoring.

Keywords

authoring, world wide web, ad hoc software, software, systematization

UVOD

Čeprav je avtorski proces v spletnem okolju podprt z vrsto programskih pripomočkov, je lahko avtor postavljen v situacijo, ki terja izdelavo dopolnilne programske opreme. V preteklih deset in več letih izdelave učnih materialov za izobraževanje preko svetovnega spleta je bilo več takšnih situacij. Od situacije do situacije so nastajali programi, ki so omogočali razreševanje nastalih problemskih situacij (npr. pretvarjanje formatov dokumentov, slogovno urejanje, urejanje obsežnejših preglednic, povezovanje, indeksiranje, enostavni animirani prikazi). Z leti se tako nabere kar nekaj programov, ki so izdelani zgolj za preseganje trenutne problemske situacije z viri, ki so v dani situaciji na razpolago. Ti programi *nimajo* večine atributov kakovostnih programov. Uporablja jih lahko le njihov avtor. Označimo jih kot *ad hoc* programi.

Namen avtorskega procesa ni izdelava programske opreme, vendar pa v tem procesu nastopajo situacije, ki jih je smiselno preseči z dopolnilno programsko opremo. Ker imajo situacije lastnost ponovljivosti, se lahko odločimo za sistematizacijo te programske opreme.

Zakaj sistematizacija? S sistematizacijo želimo povečati kakovost *ad hoc* programov. Dokler pojmujemo *ad hoc* program zgolj kot sredstvo za premostitev posamezne problemske situacije, je tak program uporaben pripomoček. S časom, ko se izoblikuje nabor *ad hoc* programov in se začne izražati ponovljivost problemskih situacij, pa situacijo obremenjujejo:

- slaba preglednost nabora *ad hoc* rešitev,
- otežena uporaba nesistematiziranih rešitev pri iskanju rešitev za nove problemske situacije,
- podvajanje dela v izdelavi *ad hoc* programov (ponov-

no iskanje rešitev za že rešene delne probleme),
- otežena uporaba *ad hoc* programov po preteku časa (potrebno je ponovno učenje).

S sistematizacijo želimo poiskati rešitev, ki bi omogočala reševanje sedanjih problemov, za katere smo že izdelali programe, in reševanje prihodnjih problemov. Zato sistematizacijo zasnujemo namesto na seznamu problemov na seznamu kategorij problemov, v katere lahko uvrstimo že rešene in pričakovane probleme:

- slogovna ureditev dokumenta,
- preverjanje in pretvarjanje formata dokumenta,
- dopolnjevanje dokumenta (npr. zaznamki, poveze, indeksi),
- preureditev zaporedja slik v dokumentu v enostaven interaktiven prikaz,
- izvlečenje podatkov iz besedila dokumenta in ureditev podatkov v strukturiran zapis,
- pretvorba nabora spletnih strani v dokumente in povezovanje v enovit dokument (npr. učbenik),
- pretvorba nabora dokumentov v spletne strani in povezovanje v splet,
- razdelitev dokumenta na več dokumentov, pretvorba v spletne strani in povezovanje v splet.

Razčlenitev problemov na delne probleme razkrije podvajanje (npr. pretvorba dokumenta v spletno stran): podvajanje delnih problemov obeta zmanjšan obseg dela, saj lahko za enake delne probleme uporabimo isto rešitev. Iskanje rešitev za probleme sloni na razčlenjevanju.[1]

RAZČLENJEVANJE

Z razčlenjevanjem prevedemo problem v soodvisne delne probleme, za katere že poznamo postopke reševanja ali pa jih znamo poiskati. Soodvisnost poveže delne probleme v strukturo, ki ima navzven enake lastnosti kot problem, za katerega iščemo rešitev. V to strukturo uvedemo dodatno soodvisnost, ki delne probleme uredi v *zaporedja*. Tej strukturi priredimo izomorfno strukturo operacij, s katerimi rešujemo delne probleme prvotne strukture. Ta struktura določa v našem primeru potek reševanja problema in jo uporabimo kot osnovo za oblikovanje sistema.

Izvajanje operacije v strukturi operacij je pogojeno s *stanjem*, ki ga določijo tej operaciji predhodne *operacije*. Trenutno stanje pogojuje operacijo, ki naj se izvede. Takšen pogled nakaže možno osnovno ureditev sistema, s katerim rešujemo problem: *krmiljenje nabora operacij*. Krmiljenje povzema prehajanje stanj in tako določa vrstni red izvajanja operacij, ki so v naboru operacij.

Ker je bila *koda*[2] za operacije izdelana že pred sistematizacijo, je postopek razčlenjevanja podan le toliko, da lahko izluščimo izhodiščno shemo sistema za izvajanje operacij. Dejansko pa je bila pred oblikovanjem sistemov za reševanje problemov ponovno izvedena razčlenitev operacij z namenom preveriti ustreznost operacij v *ad hoc* programih. Ta razčlenitev določa vsega deset kategorij operacij:

- *izvlečenje dokumenta*[3] (iz nabora dokumentov),
- *zamenjava dokumenta* (v naboru dokumentov),
- *izvlečenje elementa*[4] (iz dokumenta),
- *zamenjava elementa* (v dokumentu),
- *razdelitev dokumenta* (na elemente),
- *sestavljanje dokumenta* (iz elementov),
- *brisanje elementa* (v dokumentu),
- *vstavljanje elementa* (v dokument),
- *dodajanje elementa* (k dokumentu),
- *operacija z elementom* (sprememba elementa).

Posamezni kategoriji operacij lahko pripada ena operacija, dve ali več.

Nabor kategorij operacij bi bilo mogoče še nekoliko zmanjšati (npr. operacijo *brisanje elementa* je moč nadomestiti z operacijo *zamenjava elementa* s praznim nizom), vendar želimo nabor kategorij, ki nam je tudi pomensko blizu.

Večino že izdelane kode za izvajanje operacij je bilo mogoče v nekoliko spremenjeni obliki uporabiti tudi v sistematiziranih sistemih.

OBLIKOVANJE SISTEMOV

Z razčlenjevanjem smo določili tudi osnovno obliko sistema, s katerim lahko rešujemo problem: *krmiljenje nabora operacij*. V nadaljevanju podajamo nekaj oblik sistemov, ki so izpeljani iz osnovne oblike sistema.

Program

Program ima enake lastnosti, kot jih določa osnovna razčlenitev sistema. Ena od oblik izvedbe sistema je lahko program (npr. *ad hoc* program), ki ima takšno krmiljenje in operacije, da jih sistem, s katerim izvajamo program, zna izvajati.

Posplošitev operacij

Zamislimo si *zaporedje* operacij, ki smo jih določili z razčlenitvijo kategorij problemov. To zaporedje operacij naj izvaja za avtorja smiselne operacije (npr. izdelava zaznamkov, izdelava povezav, izdelava indeksa). Vsakemu

delnemu zaporedju operacij (npr. izdelava zaznamkov) priredimo *posplošeno operacijo* (npr. izdelava zaznamkov) in *ukaz*, ki določa to operacijo. Na tak način oblikujemo nabor posplošenih operacij. Oblikujemo pa tudi nabor ukazov, ki določajo natanko te operacije. S tako določenimi ukazi lahko izdelujemo *programe*,[5] ki vsebujejo le operacije iz nabora posplošenih operacij:

- Sestavljanje posplošenih operacij v program se izvaja z operacijami smiselnimi za avtorja, (npr. izdelava indeksa, izdelava povezav).
- Število ukazov (posplošene operacije) v programu je manjše.
- Posplošene operacije so sestavljene iz operacij, dobljenih z razčlenjevanjem.
- Program je lažje razumljiv (manj ukazov in smiselne operacije z vidika avtorskega procesa).

S posplošitvijo operacij izdelavo programa prenesemo iz domene izdelave kode v domeno sestavljanja ukazov.

Program, ki je izdelan za reševanje enega problema, se lahko včasih s spremembo *vzorcev*[6] [1] uporabi za reševanje drugih problemov (npr. program za oblikovanje zaznamkov, indeksa in povezav do preglednic se lahko uporabi tudi za enake operacije z vsebinskimi celotami, primeri in vajami).

Par: program in procesor

Želimo, da obravnava računalnik programe enako kot dokumente, da jih zna tudi razčleniti na ukaze in te ukaze izvajati. Vsak dokument – program naj vsebuje povezavo do procesorja,[7] ki je sposoben izvajati ukaze v dokumentu – programu. Za uporabnika pa so lahko takšni programi oblikovani kot dokumenti XML [2], kar omogoča prijaznejšo obliko prikaza.

V našem primeru določata procesor *koncept procesiranja* in *nabor operacij*, ki jih procesor izvaja. Konceptov procesiranja je veliko. Navajamo nekaj primerov:

- Avtor izvaja program z naborom dokumentov. Procesor poskrbi za izbiranje dokumentov. Program se izvaja z enim dokumentom naenkrat (slika 1).
- Procesor prekine izvajanje ukaza, ki zahteva vnos podatkov. Ko avtor vnese podatke, se izvajanje nadaljuje.
- Pri pretvarjanju dokumentov v druge formate so možne napake ali pomanjkljivosti v strukturi izvornih dokumentov. Pred pretvarjanjem izvaja procesor preverjanje strukture dokumenta. Če najde napako, izpiše tisti element dokumenta, ki vsebuje napako, in čaka na popravek. Ko avtor vnese popravek, procesor nadaljuje preverjanje.
- Oblikovanje in sestavljanje spletnih strani v dokument, prirejen za tiskanje, zahteva več programov. Procesor preoblikuje spletne strani (npr. vsebine, vprašanja, odgovori) z različnimi programi. Program za upravljanje in sestavljanje spletnih strani pa je skupen. Procesor zaustavlja izvajanje programa za upravljanje in sestavljanje spletnih strani in aktivira izvajanje enega od programov za preoblikovanje spletnih strani glede na vrsto spletne strani. Nato ponovno aktivira program za sestavljanje spletnih strani.

Več konceptov procesiranja lahko združimo v eni konfiguraciji procesorja (npr. prvi, drugi in tretji koncept).

Par *program in procesor* lahko posplošimo podobno, kot smo posplošili zaporedje operacij, saj prav program in procesor v našem primeru določata zaporedje operacij. Tako priredimo paru program in procesor *ukaz*, v procesor, ki izvaja ta ukaz, pa vgradimo par program in procesor enako kot *operacijo*. Par program in procesor lahko vgradimo tudi v kodo operacije. Tako lahko oblikujemo tudi konfiguracijo procesorja, ki omogoča četrti koncept procesiranja, naveden med primeri za koncepte procesiranja. Koda operacije z vgrajenim parom program in procesor najprej preveri izpolnjenost pogojev za izvajanje programa. Če so ti pogoji izpolnjeni, aktivira procesor in s tem izvajanje programa, sicer pa ne.

```
1.  izvlečenje dokumenta iz nabora dokumentov
2.  [od: 1] pretvorba zapisa programa v seznam ukazov in izbira ukaza za
    izvajanje
    [od: 4] izbira naslednjega ukaza za izvajanje (sekvencer)
3.  razčlenitev ukaza na elemente in izvlečenje podatkov (tudi imena
    operacije) (dekodirnik)
4.  izbiranje in izvajanje operacije (operacijska enota)
ponavljanje korakov 2 do 4 za vsak preostali ukaz v seznamu ukazov
5.  zamenjava dokumenta v naboru dokumentov
ponavljanje korakov 1 do 5 za vsak preostali dokument iz nabora dokumentov
```

Slika 1: Postopek v procesorju za nabor dokumentov

Osnovna funkcija procesorja je izvajanje programa, čeprav ga privzeto obravnavamo kot predmet (strojno opremo).

Primer 1: Izdelava zaznamkov, povezav in indeksov

Če pripravljamo npr. tri vrste spletnih tečajev, potem lahko z enim programom in procesorjem izdelamo npr. zaznamke, povezave in indekse za vse spletne tečaje in za vse verzije spletnih tečajev (ena verzija spletnega tečaja vsebuje več sto zaznamkov, do tisoč povezav in več deset indeksov). Obvladovati je treba tisoče elementov. Tega dela ne želimo opravljati ročno.

Programi in procesorji, vstavljeni v kodo

Izhajajmo iz kategorije programov, ki uredijo spletne strani in jih sestavijo tako, da je sestavljena oblika primerna

za tiskanje. Koda, s katero bi omogočili veliko možnih oblik za tiskanje, je zahtevna, kljub temu pa je osnovni scenarij urejanja in sestavljanja spletnih strani bolj ali manj enak.

Za urejanje spletnih strani in sestavljanje izdelamo skupno kodo, ker v tej kodi ne pričakujemo sprememb. Za preoblikovanje spletnih strani pa izdelamo programe in procesor, ker pričakujemo, da jih bo treba spreminjati.

V *splošnem* ostane pri programih in procesorjih, vstavljenih v kodo, osnovna koda nespremenjena, saj smo tiste dele kode, ki jih bo treba spreminjati, nadomestili s programi in procesorji, ki znajo te programe izvajati. Programe in procesorje izdelamo le za tisto varianto, ki jo tudi potrebujemo, in tudi variante uvajamo tedaj, ko jih potrebujemo. Delamo torej le to, kar v dani situaciji potrebujemo.

Primer 2: Izdelava učbenikov za spletne tečaje

To obliko sistema smo uporabili za izdelavo učbenikov za spletne tečaje iz naborov spletnih strani (vsebine, vprašanja, odgovori). Potrebni so trije programi in en procesor, ki izvaja vse tri programe. Za pretvarjanje formatov (npr. v format RTF[8] in format PDF[9]) pa uporabljamo komercialne programe.

Programi in procesor

Z ukazi za operacije iz nabora posplošenih operacij lahko zgradimo vrsto programov. Če je nabor teh operacij nespremenljiv, lahko zadošča en procesor, ki lahko združuje več konceptov procesiranja, za izvajanje katerega koli programa. To obliko sistema smo doslej največ uporabljali za pretvarjanje formatov dokumentov in za slogovno urejanje. Pri takšnih pretvorbah včasih izvajamo tudi delno transformiranje strukture dokumentov, ki jih pretvarjamo v druge formate. V manjši meri smo uporabili ta sistem tudi za preoblikovanje kode (urejanje in zamenjevanje enakovrednih struktur).

Primer 3: Pretvarjanje formata

Pred nekaj leti smo z uporabo vzorcev struktur na spletnih straneh izvedli pretvorbo spleta ob hkratni delni transformaciji strukture spletnih strani (preko sto preglednic je bilo npr. preoblikovanih v različne sezname v formatu DocBook[10] [3]). Ob hkratnem učenju formata DocBook smo transformacijo spleta v učbenik izvedli v nekaj dneh. Obseg učbenika v formatu DocBook je bil okrog 240 strani formata A4. Tak program se lahko uporabi za transformiranje vseh spletov, ki imajo značilnosti, določene z vzorci struktur.

Velikokrat je treba dokumente pred pretvarjanjem preveriti in urediti, lahko pa so potrebne tudi strukturne prilagoditve. Za kakšen strežniški format pa program za pretvorbo ni na razpolago (od uporabnika se namesto tega pričakuje, da bo izpolnil obrazce in tako prenesel podatke na strežnik).

Programi in procesorji

Nabor operacij smo prilagodili potrebam avtorskega procesa. Zaradi tega je lahko število posplošenih operacij večje, kot število operacij, ki jih dobimo z razčlenjevanjem. Procesor, ki bi izvajal vse te operacije, bi bil obsežen in morda manj zanesljiv. Namesto enega (istega) procesorja vsakemu *programu* priredimo *procesor* tako, da v program vgradimo *povezavo* do tistega procesorja, ki je sposoben program izvajati. Pred izvajanjem programa se najprej namesti procesor zanj.

Koncept programi in procesorji sestavljajo:

- *nabor operacij* (posplošene operacije),
- *nabor ukazov* (za nabor operacij),
- *nabor programov,*
- *nabor procesorjev* (za nabor programov).

Program sistema *programi in procesorji* je le *zaporedje ukazov*. Vsak ukaz določa operacijo za izvajanje. Kodo, ki izvaja operacijo, pa dopolnimo s *strojno čitljivim zapisom* (podobno kot komentar v kodi), ki omogoča, da se *samodejno* oblikuje:

- *ukaz* za operacijo,
- *obrazec*, v katerega vnesemo parametre ukaza,
- *koda* za poizkusno izvajanje operacije.

Sestavljanje programa temelji na rezultatu vsake operacije:

> [rezultat operacije je pravilen] ukaz, dopolnjen z vrednostmi parametrov, ki se doda programu

Sestavljanje procesorja se izvaja za vsak program. Nabor operacij procesorja se tvori samodejno:

> [rezultat je pravilen, procesor ne pozna operacije] oblikuje se koda operacije in vgradi v kodo procesorja

Tako program kot procesor se namestita ob prvi izbiri operacije. S sestavljanjem programa pa se dopolnjujeta. Namestiti je mogoče kateri koli program v naboru programov, ki vsebuje tudi prazni program (ne vsebuje ukazov). Procesor, ki se namesti, je določen s povezavo v programu. Če je program prazen, lahko uporabnik izbere kateri

koli procesor (npr. v načrtovanem programu namerava uporabiti predvsem operacije izbranega procesorja). Procesor za že izdelane programe se lahko dopolni z operacijami za izvajanje novih programov. Če se namesti prazni program, se lahko namesti tudi prazni procesor. Prazni procesorji so vse tiste konfiguracije, ki ne izvajajo nobene operacije.

IZVEDBA SISTEMA PROGRAMI IN PROCESORJI

Odločili smo se za izvedbo v spletnem okolju z uporabo skriptnega jezika PHP [4]. Tudi drugi prej opisani sistemi so izvedeni na enak način. Spletno okolje nas ne omejuje na dokumente v formatih HTML [5] ali XML.

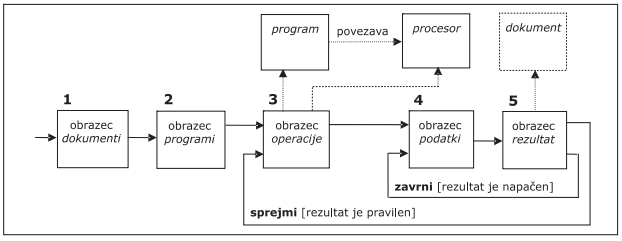
Izdelava programa in procesorja

Program sestavljamo z vpisovanjem podatkov v obrazce, ki se tvorijo samodejno. Pri sestavljanju programa smo razbremenjeni mnogih podrobnosti (slika 2 in slika 3).

```
1. izbiranje dokumenta v naboru dokumentov
2. izbiranje programa v naboru programov
      1. izbiranje procesorja, če je izbran prazni program
3. izbiranje operacije v naboru operacij
4. vnos podatkov v obrazec za podatke
5. izvajanje operacije in prikaz rezultata
izvajanje se nadaljuje s korakom 3, če je rezultat pravilen
izvajanje se nadaljuje s korakom 4, če je rezultat napačen
```

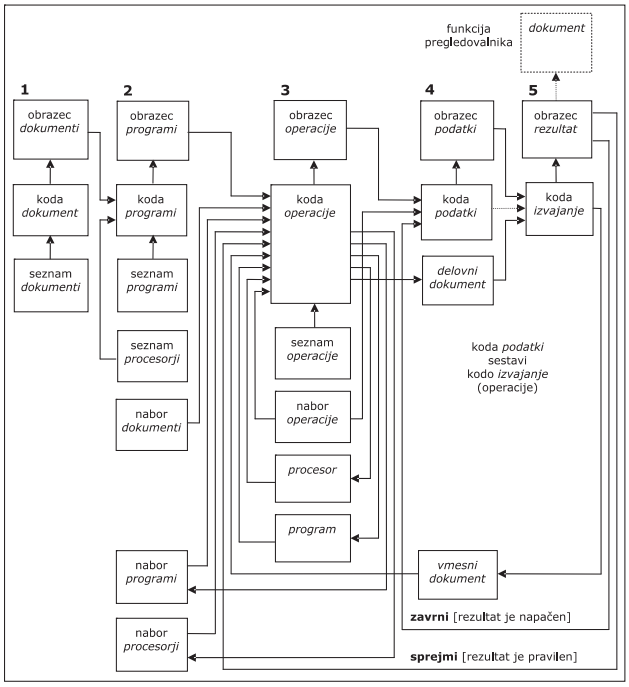
(Če se izvajanje nadaljuje s korakom **3**, se program dopolni z ukazom za operacijo, procesor pa z operacijo, če ta vanj še ni vgrajena.)

Slika 2: Postopek sestavljanja programa



Slika 3: Shema sestavljanja programa

Dejanski postopek, ki se odvija ob sestavljanju programa, je bolj zapleten, kot tisti, ki ga izvaja uporabnik. Več postopkov se odvija samodejno v ozadju; uporabnik ima dostop le do rezultatov, ki jih določijo ti postopki (slika 4).



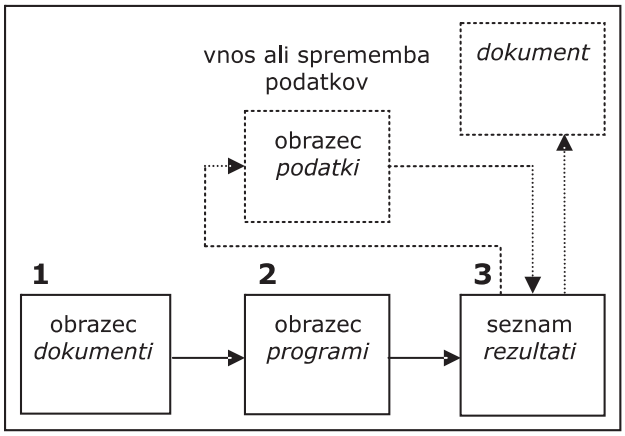
Slika 4: Shema sestavljanja programa in procesorja

Izvajanje programa v delovnem okolju

Ko sestavimo program, ga lahko avtor izvaja v delovnem okolju (slika 5). Namestitev procesorja, ki ga določa povezava v izbranem programu, je samodejna. Avtor izbere le dokumente in program, s katerim želi obdelati te dokumente (slika 6).

```
1. izbiranje dokumentov v naboru dokumentov
2. izbiranje programa v naboru programov
3. izvajanje programa s pripadajočim procesorjem
      1. samodejna namestitev procesorja
      2. izvajanje programa
   ponavljanje koraka 3.2 za vsak v koraku 1 izbrani dokument
```

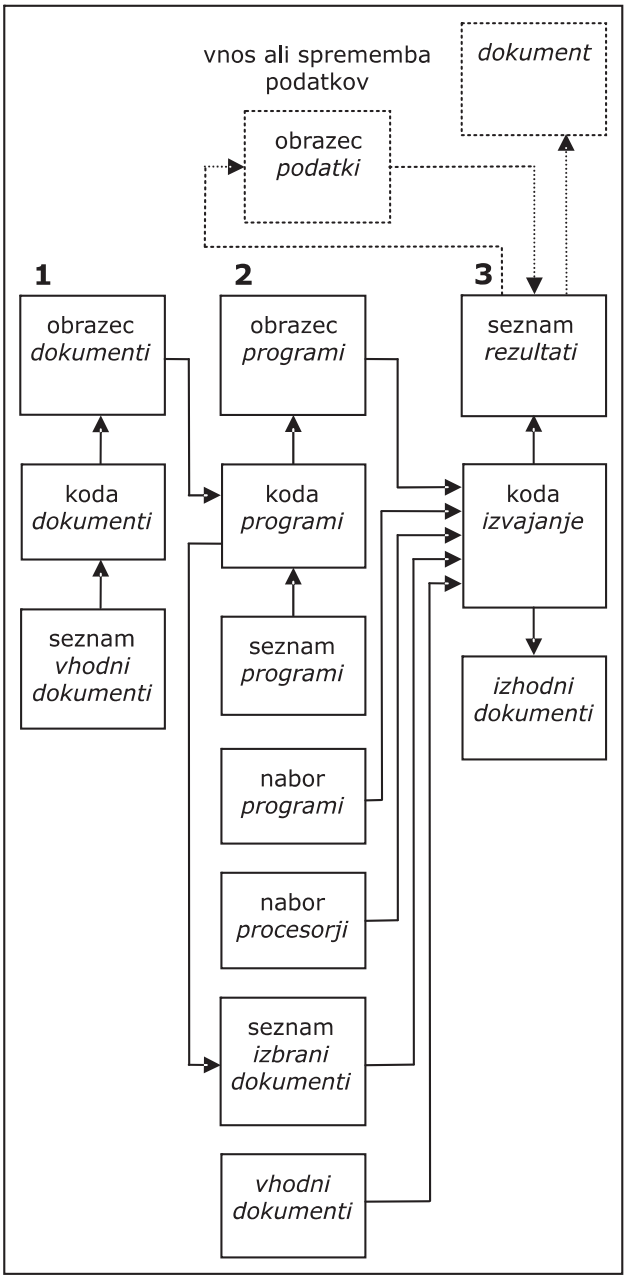
Slika 5: Postopek izvajanja programa v delovnem okolju



Slika 6: Izvajanje programa v delovnem okolju

Med izvajanjem lahko procesor od uporabnika zahteva dodatne podatke, če jih potrebuje v izvajanju.

Dejanski postopek, ki se odvija ob izvajanju programa v delovnem okolju, je nekoliko bolj zapleten od tistega, ki ga izvaja avtor (slika 7).



Slika 7: Izvajanje programa v delovnem okolju

Pregled doslej izdelanih sistemov

Doslej je bilo izdelanih:

- več *ad hoc* programov (od teh so trije zahtevnejši),
- tri verzije *programi in procesor, vstavljeni v kodo*,
- tri verzije *programi in procesor*,
- ena verzija *programi in procesorji*.

Vsa izdelana koda se uporablja v delovnem procesu (verzija *programi in procesorji* je v uvajanju v delovni proces).

Skupno je bilo z izdelanimi programi procesiranih preko 5.000 različnih dokumentov v različnih formatih. Če bi merili obseg teh dokumentov v straneh formata A4, potem bi posamezni dokument obsegal od 1 strani pa do 300 strani formata A4. Mnogi dokumenti so bili do oblikovanja končne verzije procesirani večkrat.

Primer 4: Prenos podatkov na učni strežnik

Ni si mogoče zamisliti avtorja, ki bi si želel v zaključni fazi avtorskega procesa dva ali tri dni izpolnjevati desetine obrazcev, da bi prenesel npr. 200 vprašanj z odgovori in opombami na učni strežnik. Takšen problem lahko s pretvorbo dokumentov v interni strežnikov format rešimo v nekaj minutah (pretvorba, prenos na strežnik in aktiviranje). Program za preverjanje in pretvorbo obsega vsega 9 ukazov, procesor pa izvaja 4 različne operacije. Procesor je konfiguriran tako, da omogoča nabor dokumentov in popravljanje najdenih napak v strukturi dokumentov pred pretvorbo.

Primer 5: Program za izdelavo zaznamkov, povezav in indeksov

Izdelava zaznamkov, povezav in indeksov za celotni splet tečaja se izvaja s programom in procesorjem, ki se samodejno zaustavi in izpiše obrazec za vnos podatkov (npr. imena skriptnih funkcij, imena povezav za video zapise – številčenje povezav je samodejno). Ko avtor vnese zahtevane podatke, procesor nadaljuje izvajanje. Program, ki izdela vse zaznamke, povezave, indekse in vgradi povezave do skript za celotni splet, obsega 22 ukazov, procesor pa izvaja 9 različnih operacij. Avtor vnese le okrog 30 podatkov (zgolj modificira v obrazcih vpisane podatke).

Rutinskih opravil je v okolju spletnih tehnologij veliko. S programi omogočamo, da je avtor predvsem v zaključni fazi avtorskega procesa razbremenjen teh opravil.

V najugodnejši situaciji imamo za izdelavo programa na razpolago *kodo za vse potrebne posplošene operacije in procesor*. V najmanj ugodni situaciji je treba pred izdelavo programa napisati *kodo za vse posplošene operacije in konfigurirati procesor*. Dokler ostajamo v domeni, ki smo jo določili z razčlenitvijo, imamo *kodo za operacije*, ki smo jih določili z razčlenitvijo na razpolago. Za konfiguriranje procesorja pa imamo prav tako na razpolago *kodo*

že obstoječih procesorjev. Tako je obseg dela manjši, kot če bi morali vso potrebno kodo napisati na novo.

Primerna *razčlenitev problemske situacije* in primerno *konfiguriranje sistema* omogočata znatne prihranke v obsegu potrebne kode.

ZAKLJUČEK

Odločitev za izdelavo dopolnilne programske opreme so pogojevale problemske situacije v avtorskem procesu. Začetni namen je bil z *ad hoc* programom premostiti pogoste problemske situacije. To ne pomeni, da je bil tak program napisan brez razmisleka; prvi program je bila koda za sistem *programi in procesor*, ki je nekaj let omogočal pokrivanje znatnega dela potreb. S časom so nastajale nove problemske situacije, ki so pogojevale tudi drugačne rešitve. Tako je nastajal vedno bolj obsežen nabor programov, ki ni samo nepregleden, ampak tudi vedno težje obvladljiv. Odločitev za sistematizacijo ni nova. Pogoji zanjo pa so nastopili nedavno.

S sistematizacijo lahko izboljšamo preglednost, povečamo obvladljivost, zmanjšamo obseg in podvajanje dela ter skrajšamo čas učenja za uporabo programov. S sistematizacijo lahko zagotavljamo tudi večjo zanesljivost programske opreme. Gledano v celoti pa sistematizacija ne zagotavlja vseh atributov kakovosti programske opreme. To niti ni naš namen. Programi in procesorji se uporabljajo za dokaj specifične naloge, zato v njihovo izdelavo ni mogoče vložiti toliko, kot bi vložili v programe, namenjene bolj splošni rabi.

Izdelava vseh opisanih verzij programov je bila odvisna od ocene, ali je ceneje in hitreje problem rešiti ročno ali pa napisati program za njegovo reševanje in ga rešiti strojno. Celotna sistematizacija je bila izdelana v prostem času. V prostem času je bila izdelana tudi začetna koda za sistem *programi in procesorji*. Vse aktivnosti je izvajal avtor tega prispevka. Ko je začetno preizkušanje potrdilo koncept, se je začelo postopno uvajanje sistema v avtorski proces.

Pojmovati dopolnilno programsko opremo kot še en element tekmovalnosti, bi bilo napačno. Z dopolnilno programsko opremo težimo k situaciji *win-win*[11] (avtor in založnik oz. delodajalec dobita nekoliko večji manevrski prostor, uporabniki bolj kakovosten učni material).

Nobena sistematizacija ni dokončna. Novi problemi, ki niso iz kategorije problemov, na katerih temelji sistematizacija, lahko pogojujejo drugačno sistematizacijo. Tako kot sistematiziran sistem po eni strani omogoča lažje izpolnjevanje zahtev, postavlja po drugi strani tudi omejitve. Ko začno omejitve sistema ovirati izpolnjevanje novih zahtev, je čas za razmislek in za prilagoditev ali spremembo sistema.

Reference

[1] Friedl, J. E. F. (2006). Mastering Regular Expressions, Third Edition. Farnham: O'Reilly.

[2] Evjen, B., K. Sharkey, T. Thangarathinam, M. Kay, A.Vernet, S. Ferguson (2007). Professional XML. Indianapolis, IN: Wiley.

[3] Walsh, N., L. Muellner (2006). DocBook: The Definitive Guide. O'Reilly. Dostopno/Dosegljivo na: http://www.docbook.org/tdg/en/html/docbook.html (zadnji ogled 21. 5. 2008).

[4] Olson, P. (ur.) (2008). PHP Manual. The PHP Group. Dostopno/Dosegljivo na: http://www.php.net/manual/en/index.php (zadnji ogled 21. 5. 2008).

[5] Musciano, C., B. Kennedy (2006). HTML & XHTML: The Definitive Guide, Sixth Edition. Beijing, Farnham: O'Reilly.

Opombe

1 Razčlenjevanje ni vezano na noben model načrtovanja sistemov. Koncept razčlenjevanja je neodvisen od posamezne paradigme, na kateri lahko temelji izdelava programske opreme.

2 S kodo poimenujemo v tem prispevku programsko opremo, kadar jo obravnavamo na nivoju programskega jezika.

3 Rezultat operacije je kopija dokumenta. Izvorni dokument ostane v naboru.

4 Rezultat operacije je kopija elementa. Izvorni element ostane del dokumenta. Izvlečenje izvajamo običajno z vzorcem elementa, zato je lahko rezultat operacije *izvlečenje elementa* več elementov, ki so urejeni v nabor.

5 S programom poimenujemo program, ki je zgrajen z ukazi za posplošene operacije. Kadar pa je pojem programa uporabljen v običajnem pomenu, je to razvidno iz konteksta.

6 Dostop do elementov dokumenta urejamo z vzorci, saj izvajamo operacije tudi z dokumenti, katerih elementi niso označeni. Vzorce oblikujemo z regularnimi izrazi.

7 S procesorjem označimo kodo, ki izvaja enako funkcijo kot procesor (mikroprocesor) v računalniku. Funkcija procesorja je izvajanje programa.

8 *Rich Text Format*. V tem formatu pogosto shranjujemo dokumente, izdelane s programom *Microsoft Office Word*. Oblika datotek s pripono *doc* je binarna oblika, ki je ne uporabljamo za pretvorbe formatov.

9 *Portable Document Format*.

10 *DocBook* je format v jeziku XML, namenjen tehniški dokumentaciji. Nekateri programi, npr. *OpenOfficeWriter* omogočajo tudi shranjevanje dokumentov v formatu *DocBook*. Če pa želimo vplivati na to, kako se strukture enega formata preslikajo v strukture drugega formata, potrebujemo program, ki to omogoča. Pri tem ni potrebno, da tak program preslika vse strukture izhodiščnega

formata v strukture ciljnega formata, ampak le tiste, ki so v dokumentih uporabljene.

11 V situaciji *win-win* pridobijo vsi udeleženci. Manj prijazna je situacija *I win you loose*, v kateri pridobi le eden.

Spletne povezave

- World Wide Web Consortium: http://www.w3.org/.
- PHP: Hypertext Preprocessor: http://www.php.net/.